**Exercise 1:**

**Implementing Edge Services for Routing and Filtering**

**pom.xml**

<project ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.amrutha.gateway</groupId>

<artifactId>edge-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>edge-service</name>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.4</version>

<relativePath/>

</parent>

<properties>

<java.version>17</java.version>

<spring-cloud.version>2023.0.1</spring-cloud.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-gateway</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-webflux</artifactId>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**application.properties**

spring.cloud.gateway.routes[0].id=test\_route

spring.cloud.gateway.routes[0].uri=https://httpbin.org

spring.cloud.gateway.routes[0].predicates[0]=Path=/example/\*\*

**EdgeServiceApplication.java**

package com.amrutha.gateway;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EdgeServiceApplication {

public static void main(String[] args) {

SpringApplication.run(EdgeServiceApplication.class, args);

}

}

**LoggingFilter.java**

package com.amrutha.gateway;

import org.springframework.cloud.gateway.filter.GatewayFilterChain;

import org.springframework.cloud.gateway.filter.GlobalFilter;

import org.springframework.core.Ordered;

import org.springframework.stereotype.Component;

import org.springframework.web.server.ServerWebExchange;

import reactor.core.publisher.Mono;

@Component

public class LoggingFilter implements GlobalFilter, Ordered {

@Override

public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {

System.out.println("Incoming Request: " + exchange.getRequest().getURI());

return chain.filter(exchange);

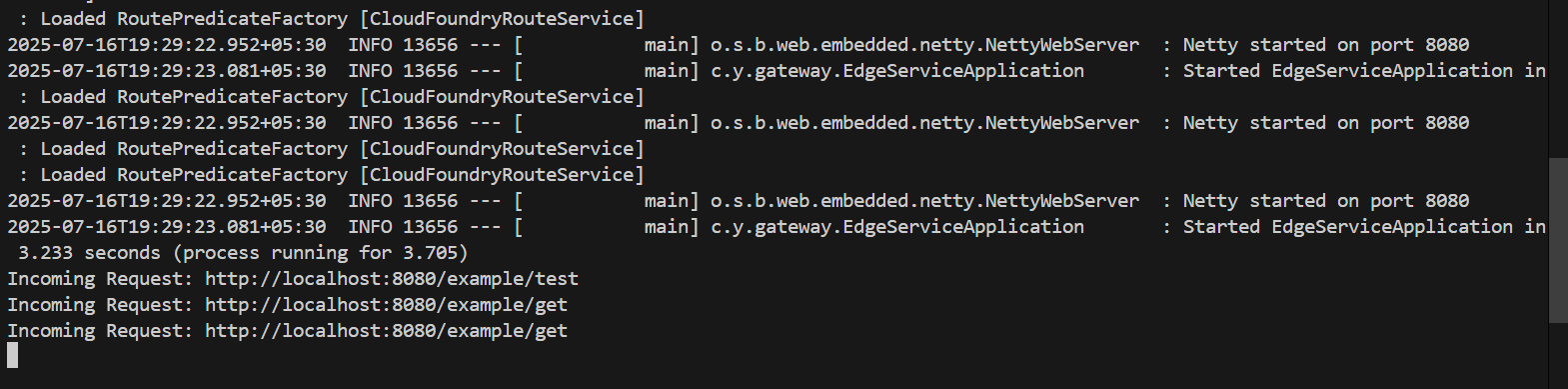
}

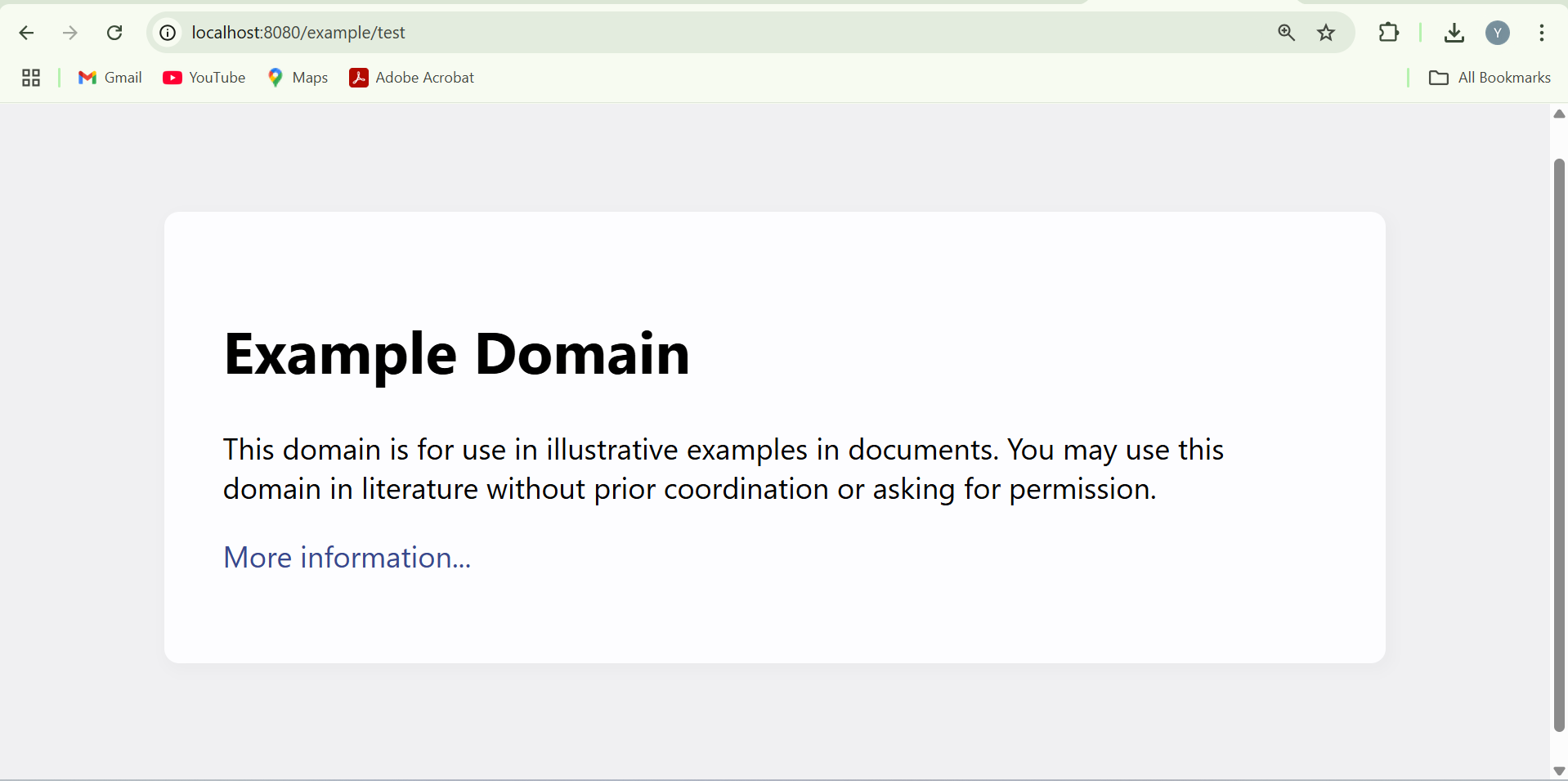
@Override

public int getOrder() {

return -1; } }

**Output:**

****

****

**Exercise 2: Load Balancing in an API Gateway**

**ExampleServiceApp1.java**

package com.amrutha;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ExampleServiceApp1 {

public static void main(String[] args) {

SpringApplication.run(ExampleServiceApp1.class, args);

} }

**ExampleController1.java**

package com.amrutha.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class ExampleController1 {

@GetMapping("/service")

public String getServiceMessage() {

return "Response from Example Service 1 (port 8081)";

}

}

**pom.xml**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

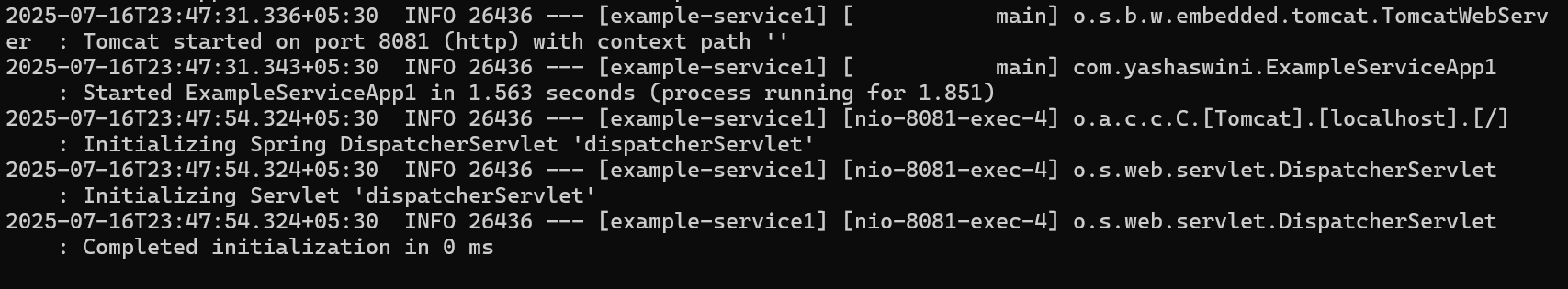
</dependencies>

**application.properties**

server.port=8081

spring.application.name=example-service

**output:**

****

****

**ExampleServiceApp2.java**

package com.amrutha;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ExampleServiceApp2 {

public static void main(String[] args) {

SpringApplication.run(ExampleServiceApp2.class, args);

}

}

**ExampleController2.java**

package com.amrutha.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class ExampleController2 {

@GetMapping("/service")

public String getServiceMessage() {

return "Response from Example Service 2 (port 8082)";

}

}

**pom.xml**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

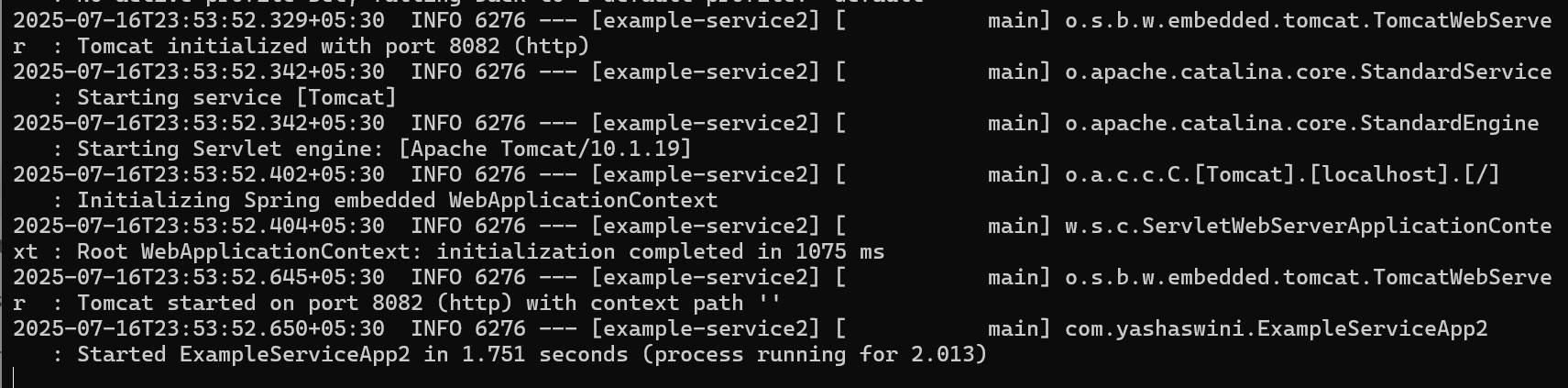
</dependencies>

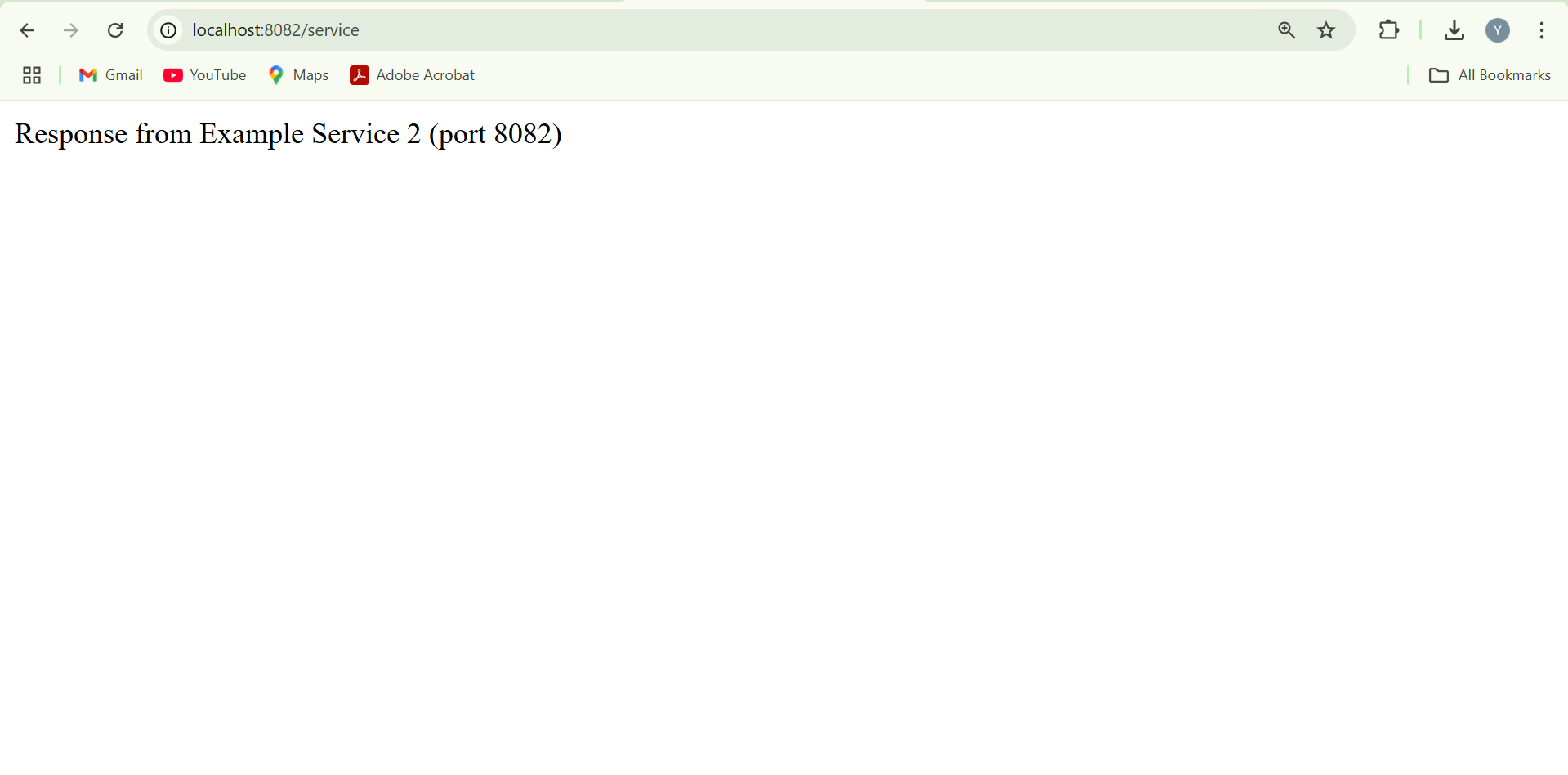
**application.properties**

server.port=8082

spring.application.name=example-service

**output:**

****

****

**GatewayApplication.java**

package com.amrutha;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.loadbalancer.LoadBalanced;

import org.springframework.context.annotation.Bean;

import org.springframework.web.client.RestTemplate;

@SpringBootApplication

public class GatewayApplication {

public static void main(String[] args) {

SpringApplication.run(GatewayApplication.class, args);

}

@Bean

@LoadBalanced

public RestTemplate restTemplate() {

return new RestTemplate();

}

}

**GatewayController.java**

package com.amrutha.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class GatewayController {

@GetMapping("/")

public String home() {

return "Welcome to the Gateway Service!";

}

}

**pom.xml**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-webflux</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-loadbalancer</artifactId>

</dependency>

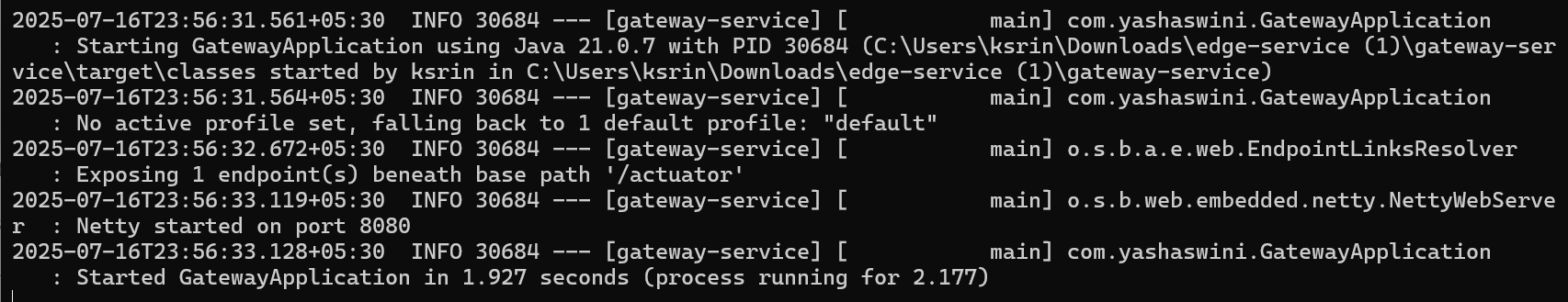
</dependencies>

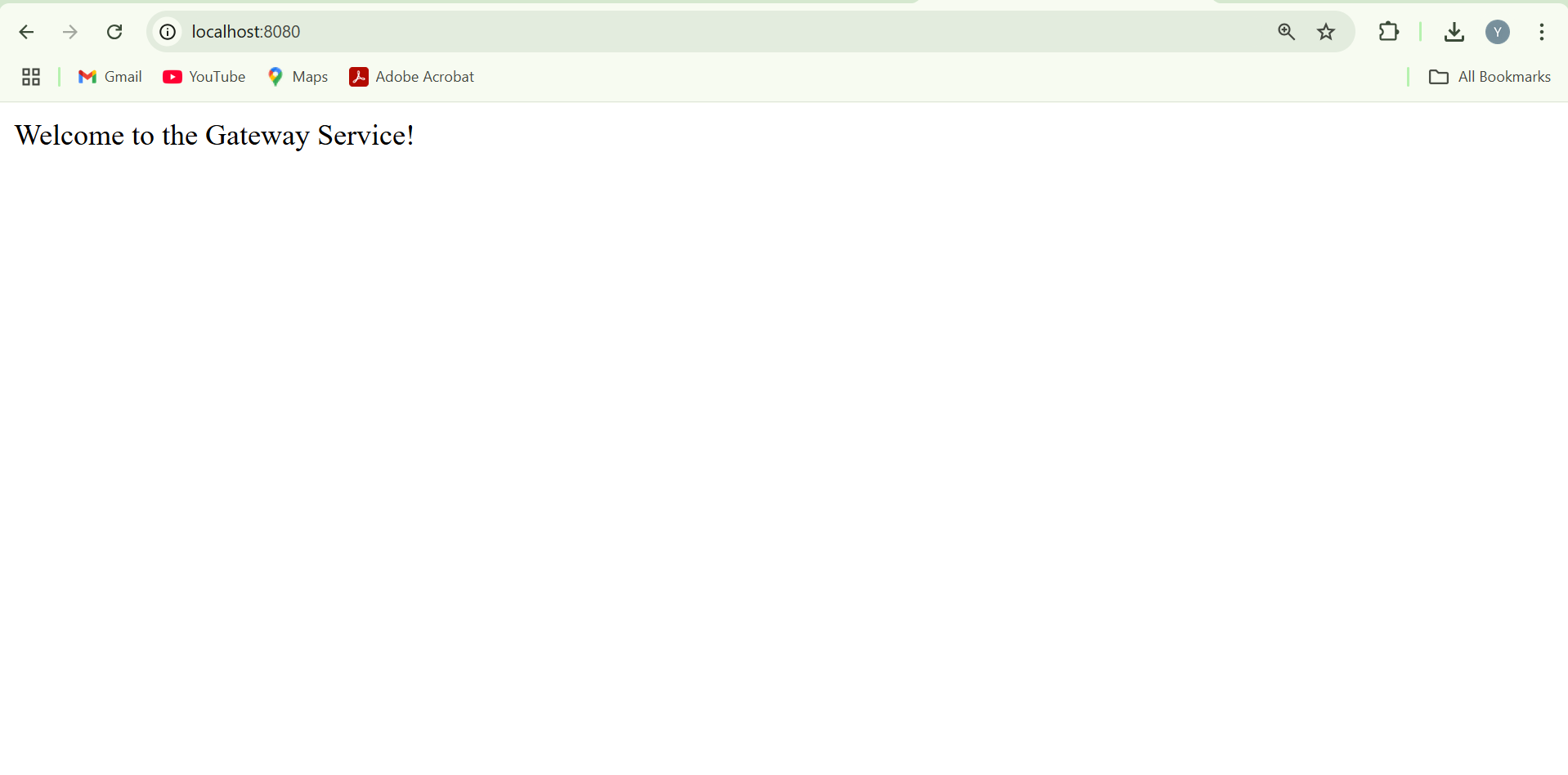
**application.properties**

server.port=8080

spring.application.name=gateway-service

**Output:**

****

****

**Exercise 3: Resilience Patterns in an API Gateway**

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.amrutha</groupId>

<artifactId>gateway-resilience</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>gateway-resilience</name>

<properties>

<java.version>21</java.version>

<spring.boot.version>3.2.4</spring.boot.version>

<spring-cloud.version>2023.0.1</spring-cloud.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-webflux</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-gateway</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-circuitbreaker-reactor-resilience4j</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**application.properties**

server.port=8080

spring.cloud.gateway.routes[0].id=example-service

spring.cloud.gateway.routes[0].uri=http://localhost:8081

spring.cloud.gateway.routes[0].predicates[0]=Path=/example/\*\*

spring.cloud.gateway.routes[0].filters[0]=CircuitBreaker=name=exampleCircuitBreaker,fallbackUri=forward:/fallback

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.slidingWindowSize=5

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.failureRateThreshold=50

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.permittedNumberOfCallsInHalfOpenState=3

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.waitDurationInOpenState=5s

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.registerHealthIndicator=true

**GatewayResilienceApplication.java**

package com.amrutha.gateway;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class GatewayResilienceApplication {

public static void main(String[] args) {

SpringApplication.run(GatewayResilienceApplication.class, args);

}

}

**FallbackController.java**

package com.amrutha.gateway.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class FallbackController {

@GetMapping("/fallback")

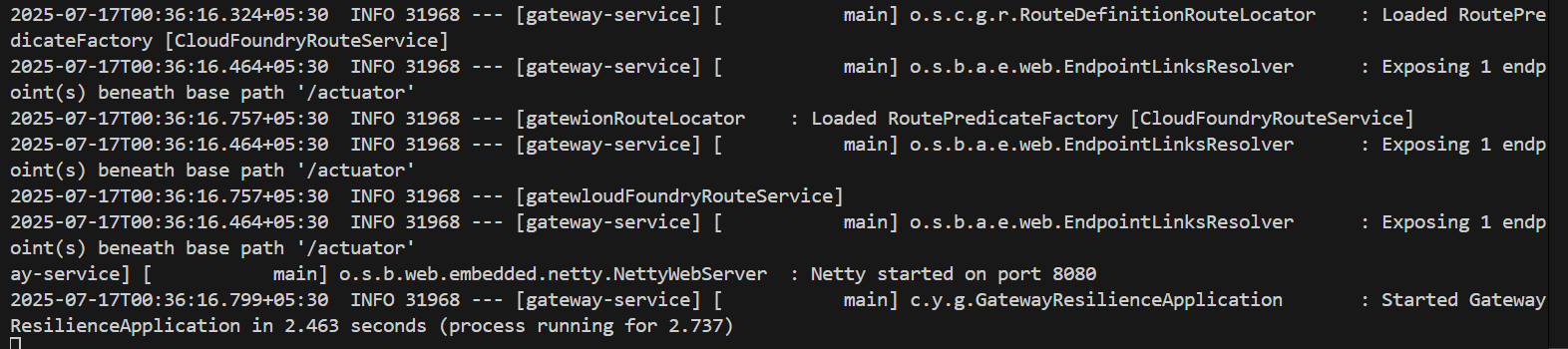
public String fallback() {

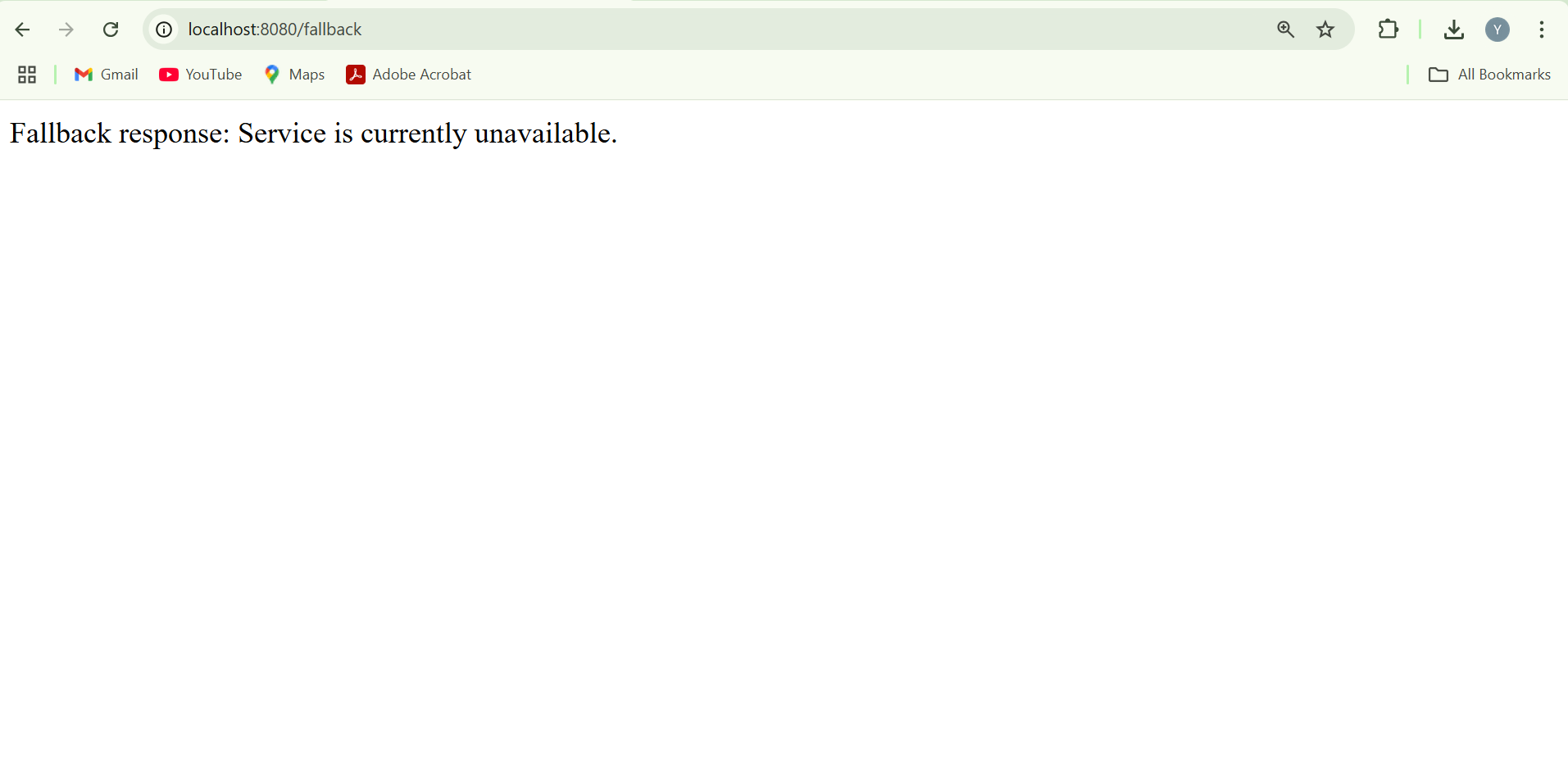
return "Fallback response: Service is temporarily unavailable.";

}

}

**Output:**

****

****

**Sample exercises on Centralized Authentication and SSO with Spring Boot 3 and Spring Cloud**

**Exercise 1: Implementing Centralized Authentication with OAuth 2.1/OIDC**

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>oauth2-login-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>oauth2-login-demo</name>

<description>OAuth2 Login Demo</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.2</version>

<relativePath/>

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-oauth2-client</artifactId>

</dependency>

</dependencies>

<properties>

<java.version>17</java.version>

</properties>

</project>

**application.yml**

server:

port: 8080

spring:

security:

oauth2:

client:

registration:

google:

client-id: YOUR\_CLIENT\_ID

client-secret: YOUR\_CLIENT\_SECRET

scope: openid, profile, email

redirect-uri: "{baseUrl}/login/oauth2/code/{registrationId}"

authorization-grant-type: authorization\_code

provider:

google:

authorization-uri: https://accounts.google.com/o/oauth2/auth

token-uri: https://oauth2.googleapis.com/token

user-info-uri: https://openidconnect.googleapis.com/v1/userinfo

user-name-attribute: sub

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

**SecurityConfig.java**

package com.example.demo.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.context.annotation.Bean;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(authorize -> authorize

.anyRequest().authenticated()

)

.oauth2Login();

return http.build();

}

}

**UserController.java**

package com.example.demo.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

import java.security.Principal;

@RestController

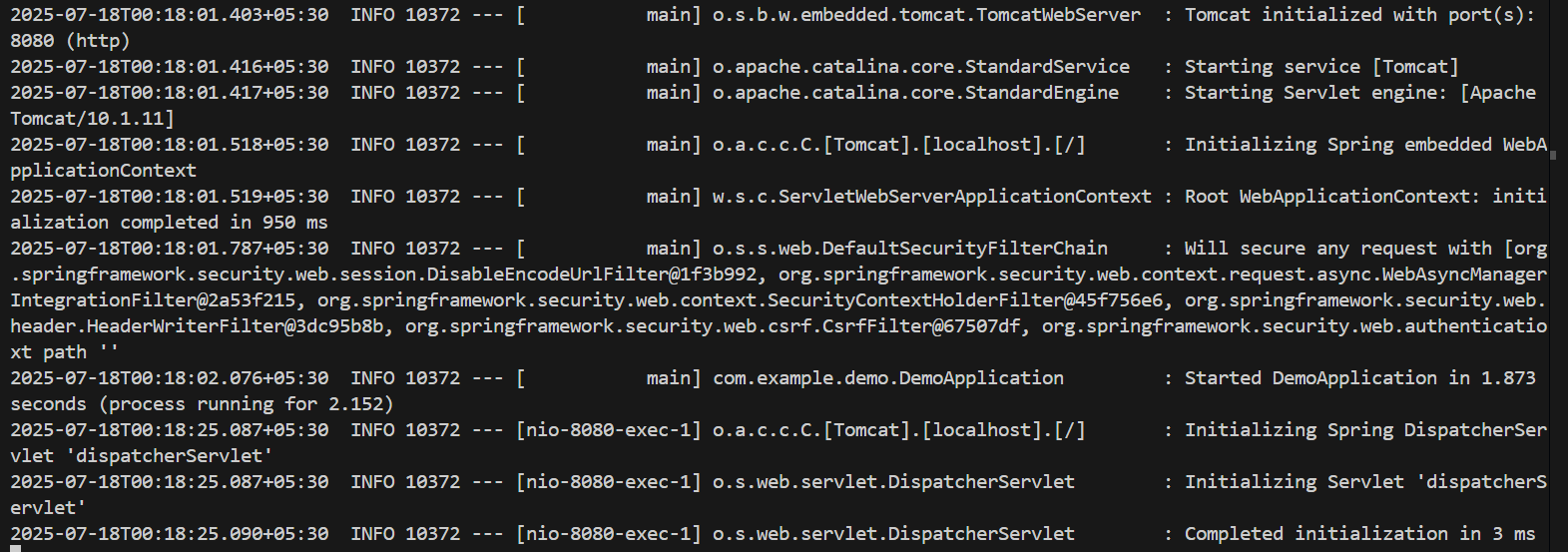
public class UserController {

@GetMapping("/user")

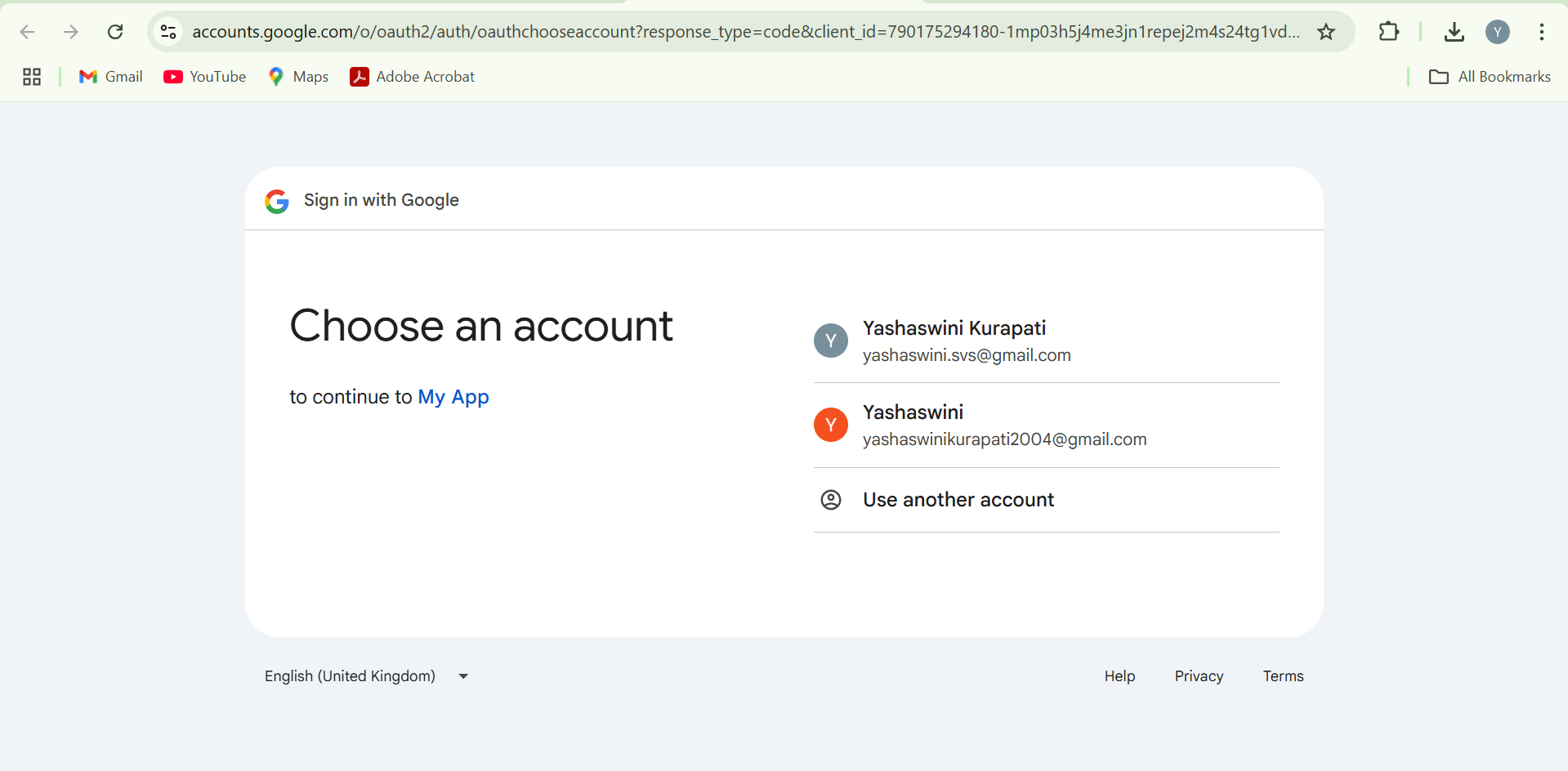
public Principal user(Principal principal) {

return principal; } }

**Output:**

****

**Open browser:** [**http://localhost:8080/user**](http://localhost:8080/user)

****

**Exercise 2: Configuring Authorization Servers and Resource Servers**

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

                             http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>

    <artifactId>oauth2-resource-server-demo</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <packaging>jar</packaging>

    <name>OAuth2 Resource Server Demo</name>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>3.2.4</version>

    </parent>

    <dependencies>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-security</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

    </dependencies>

</project>

**ResourceServerConfig.java**

package com.example.demo.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class ResourceServerConfig {

    @Bean

    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

        http

            .authorizeHttpRequests(auth -> auth

                .anyRequest().authenticated()

            )

            .oauth2ResourceServer(oauth2 -> oauth2

                .jwt()

            );

        return http.build();

    }

}

**SecureController.java**

package com.example.demo.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController

@RestController

public class SecureController {

    @GetMapping("/secure")

    public String secure() {

        return "This is a secure endpoint";

    }

}

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

    public static void main(String[] args) {

        SpringApplication.run(DemoApplication.class, args);

    }

}

**application.yml**

spring:

  security:

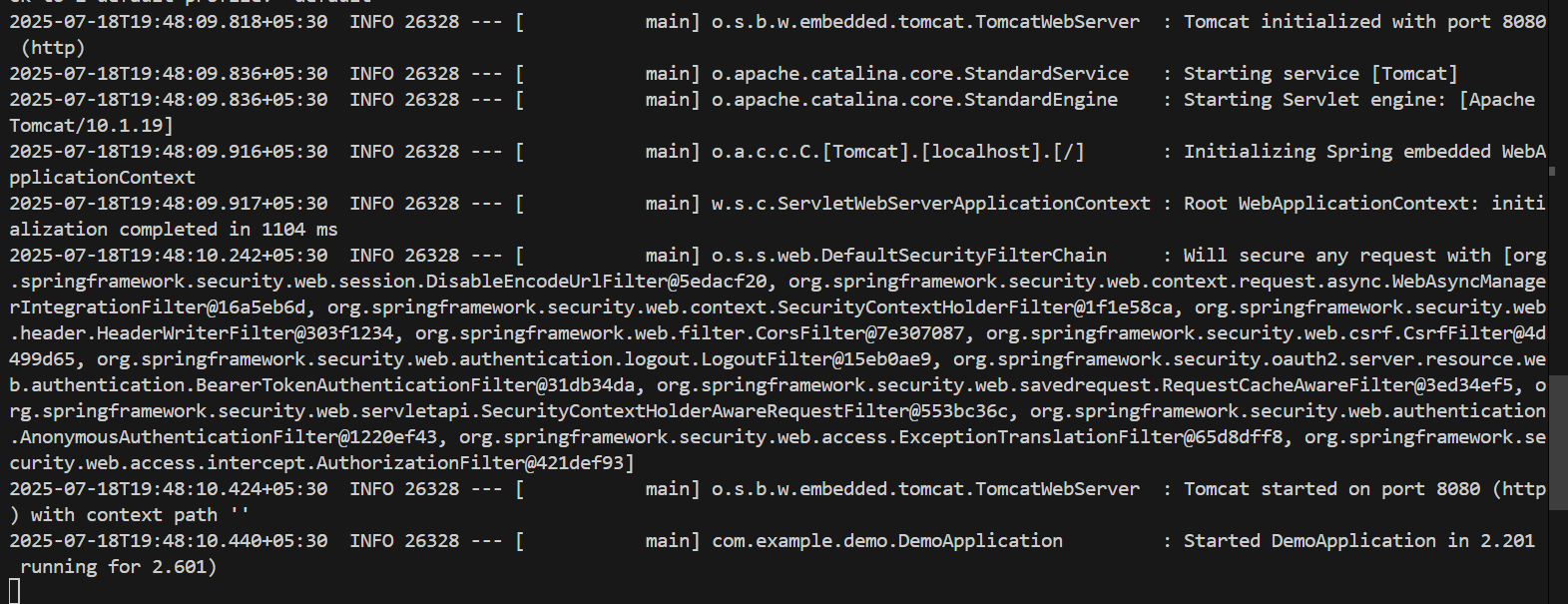
    oauth2:

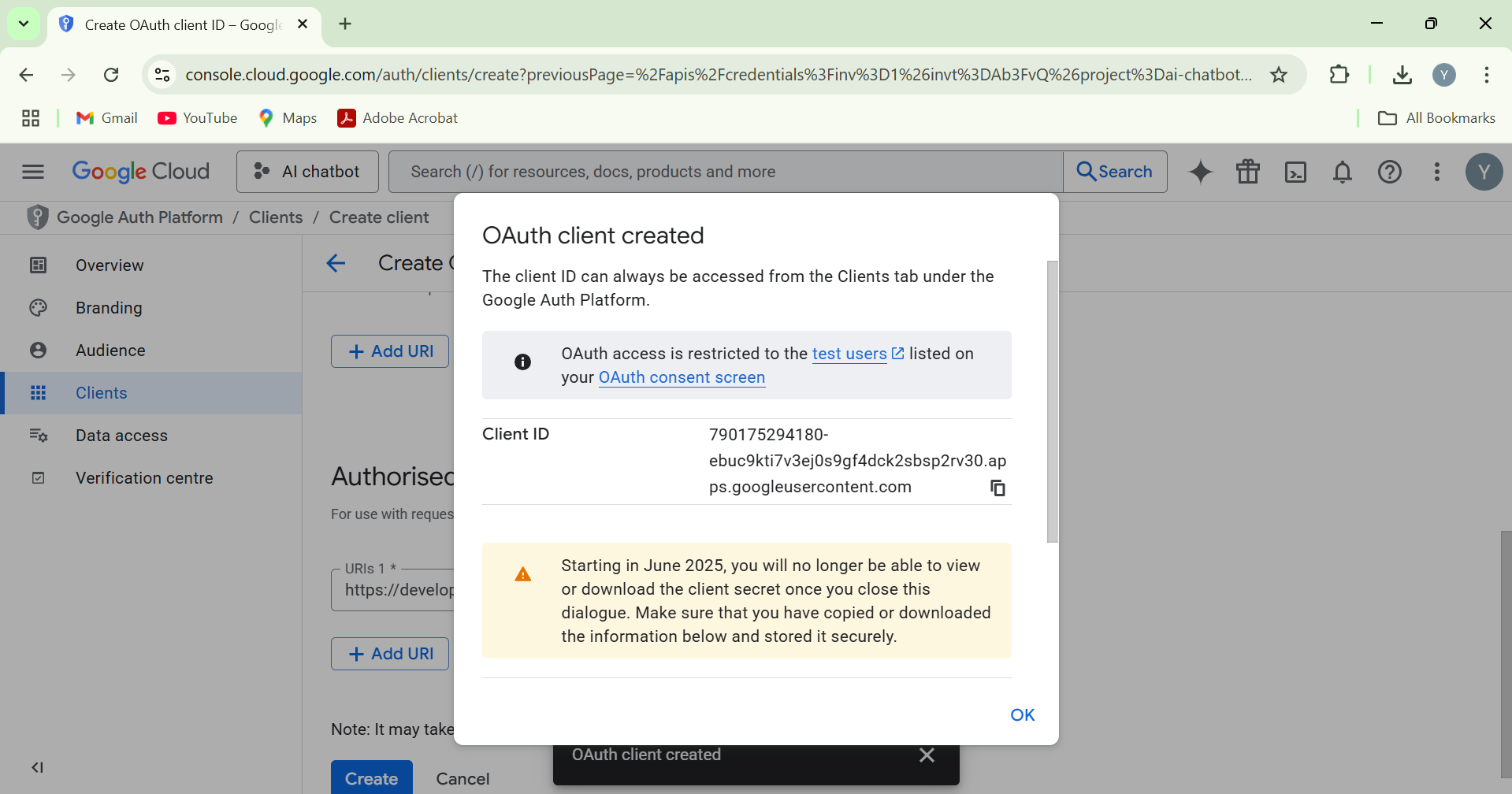
      resourceserver:

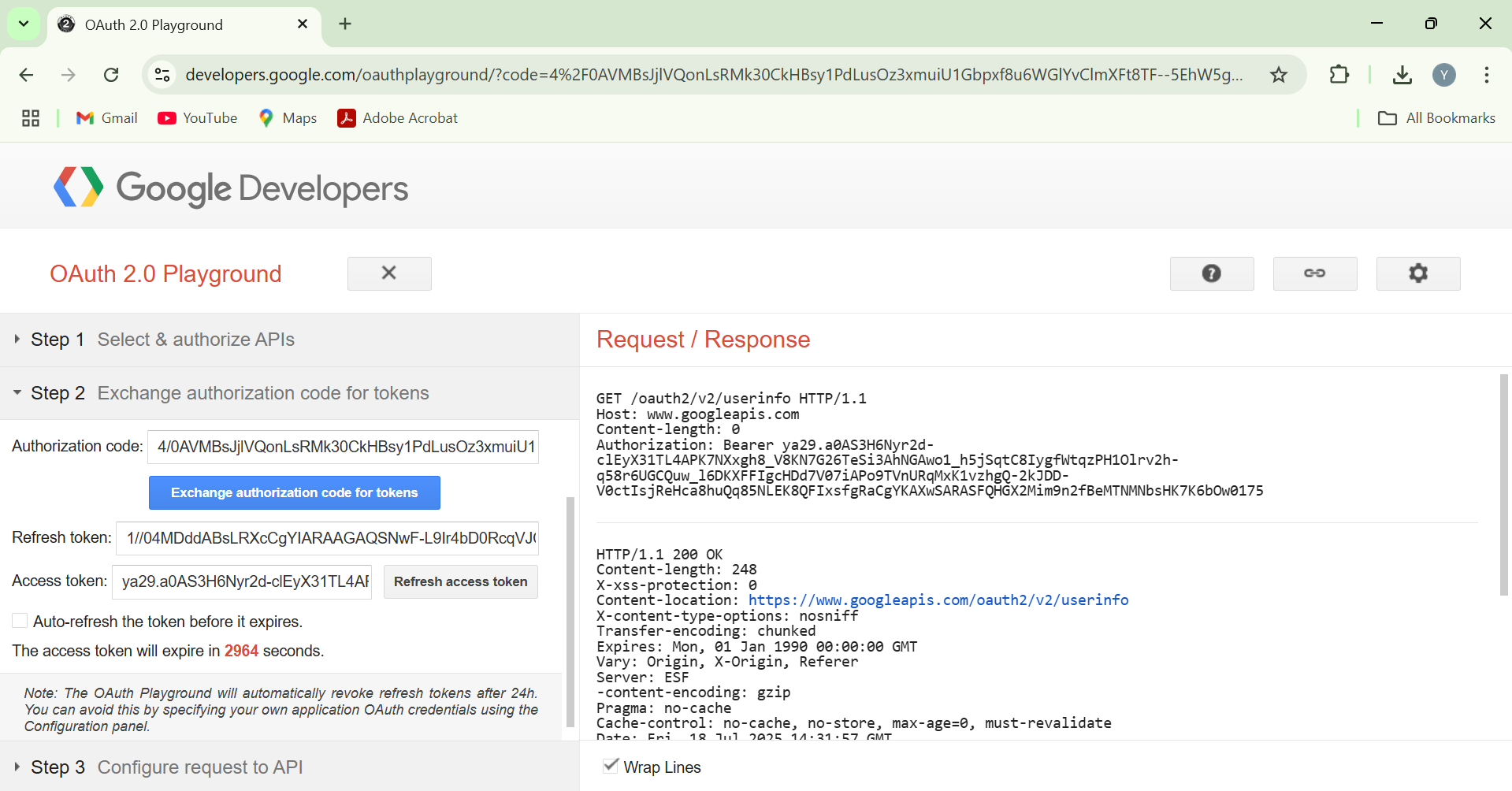
        jwt:

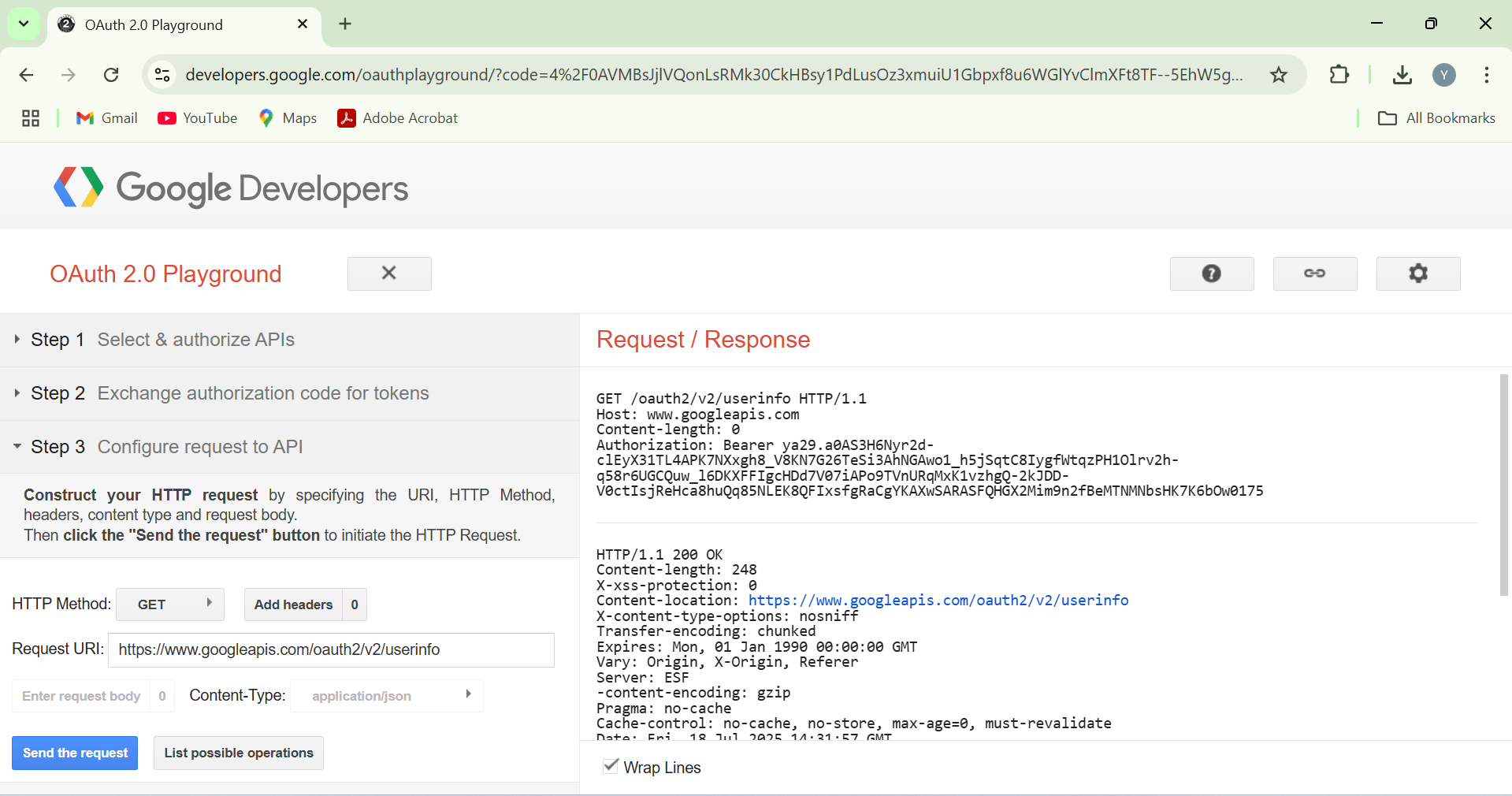
          issuer-uri: https://www.googleapis.com/auth/userinfo.email

**output:**

****

****

****

****

****

**Exercise 3: Using JSON Web Tokens (JWT) for Secure Communication**

**pom.xml**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

**application.yml**

server:

port: 8080

spring:

security:

jwt:

secret: my-secret-key

**JwtConfig.java**

package com.example.jwt.config;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.Configuration;

@Configuration

public class JwtConfig {

@Value("${spring.security.jwt.secret}")

private String secret;

public String getSecret() {

return secret; } }

**JwtTokenProvider.java**

package com.example.jwt.security;

import com.example.jwt.config.JwtConfig;

import io.jsonwebtoken.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.userdetails.User;

import java.util.\*;

@Component

public class JwtTokenProvider {

@Autowired

private JwtConfig jwtConfig;

public String createToken(String username) {

Claims claims = Jwts.claims().setSubject(username);

Date now = new Date();

Date validity = new Date(now.getTime() + 3600000);

return Jwts.builder()

.setClaims(claims)

.setIssuedAt(now)

.setExpiration(validity)

.signWith(SignatureAlgorithm.HS256, jwtConfig.getSecret())

.compact();

}

public boolean validateToken(String token) {

try {

Jwts.parser().setSigningKey(jwtConfig.getSecret()).parseClaimsJws(token);

return true;

} catch (JwtException | IllegalArgumentException e) {

return false;

}

}

public Authentication getAuthentication(String token) {

String username = Jwts.parser()

.setSigningKey(jwtConfig.getSecret())

.parseClaimsJws(token)

.getBody()

.getSubject();

return new UsernamePasswordAuthenticationToken(username, "", Collections.emptyList());

}

}

**JwtTokenFilter.java**

package com.example.jwt.security;

import jakarta.servlet.\*;

import jakarta.servlet.http.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.web.filter.OncePerRequestFilter;

import org.springframework.security.core.Authentication;

import java.io.IOException;

public class JwtTokenFilter extends OncePerRequestFilter {

@Autowired

private JwtTokenProvider jwtTokenProvider;

@Override

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain filterChain)

throws ServletException, IOException {

String token = resolveToken(request);

if (token != null && jwtTokenProvider.validateToken(token)) {

Authentication auth = jwtTokenProvider.getAuthentication(token);

SecurityContextHolder.getContext().setAuthentication(auth);

}

filterChain.doFilter(request, response);

}

private String resolveToken(HttpServletRequest request) {

String bearer = request.getHeader("Authorization");

if (bearer != null && bearer.startsWith("Bearer ")) {

return bearer.substring(7);

}

return null;

}

}

**SecurityConfig.java**

package com.example.jwt.config;

import com.example.jwt.security.JwtTokenFilter;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.\*;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.\*;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired

private JwtTokenFilter jwtTokenFilter;

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable() // for testing, disable CSRF

.authorizeRequests()

.antMatchers("/auth/\*\*").permitAll()

.anyRequest().authenticated();

http.addFilterBefore(jwtTokenFilter, UsernamePasswordAuthenticationFilter.class);

}

@Bean

public JwtTokenFilter jwtTokenFilterBean() {

return new JwtTokenFilter();

}

}

**AuthController.java**

package com.example.jwt.controller;

import com.example.jwt.security.JwtTokenProvider;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.\*;

@RestController

@RequestMapping("/auth")

public class AuthController {

@Autowired

private JwtTokenProvider jwtTokenProvider;

@PostMapping("/login")

public Map<String, String> login(@RequestParam String username) {

String token = jwtTokenProvider.createToken(username);

Map<String, String> map = new HashMap<>();

map.put("token", token);

return map;

}

}

**HelloController.java**

package com.example.jwt.controller;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/api")

public class HelloController {

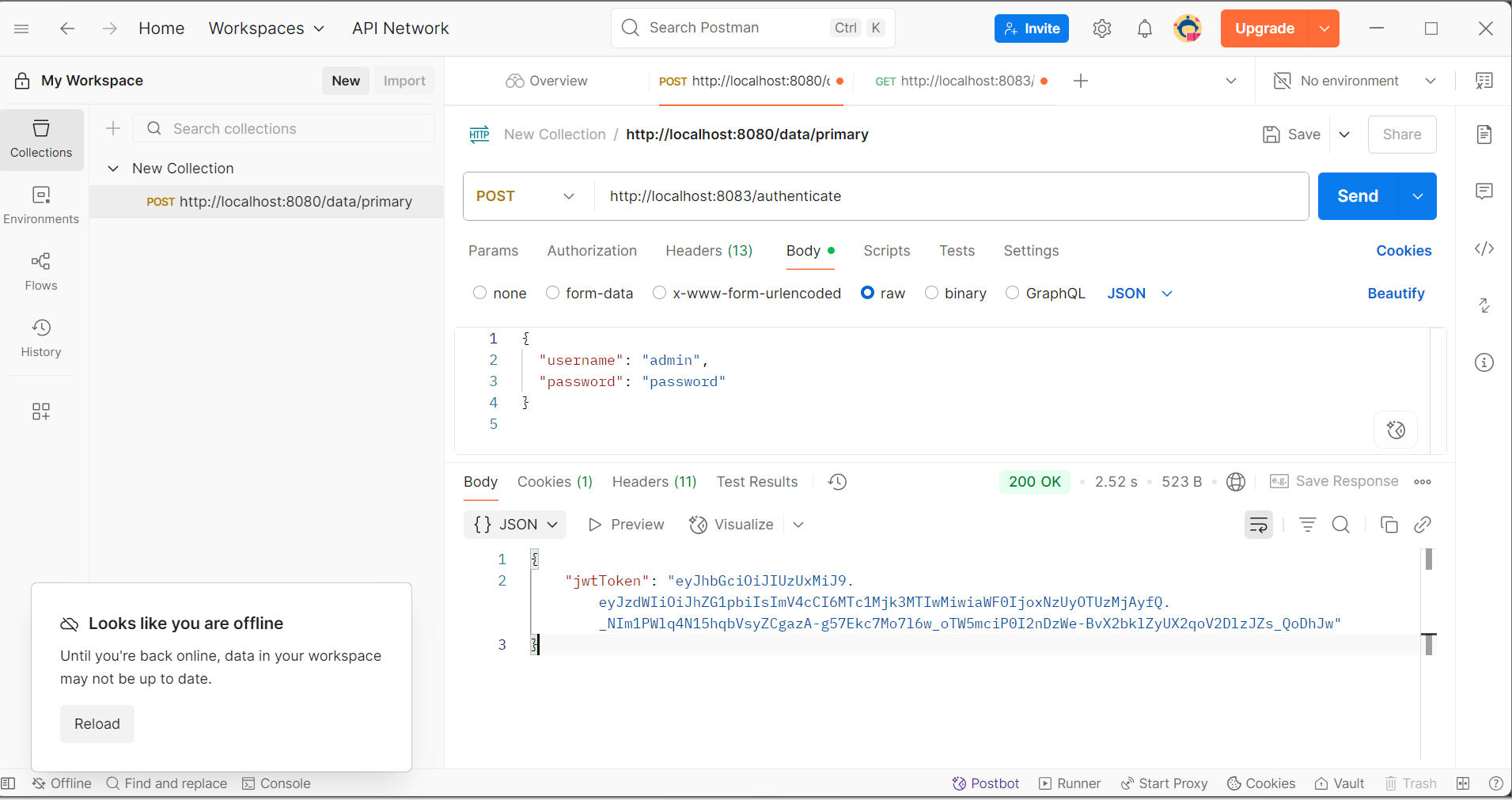
@GetMapping("/hello")

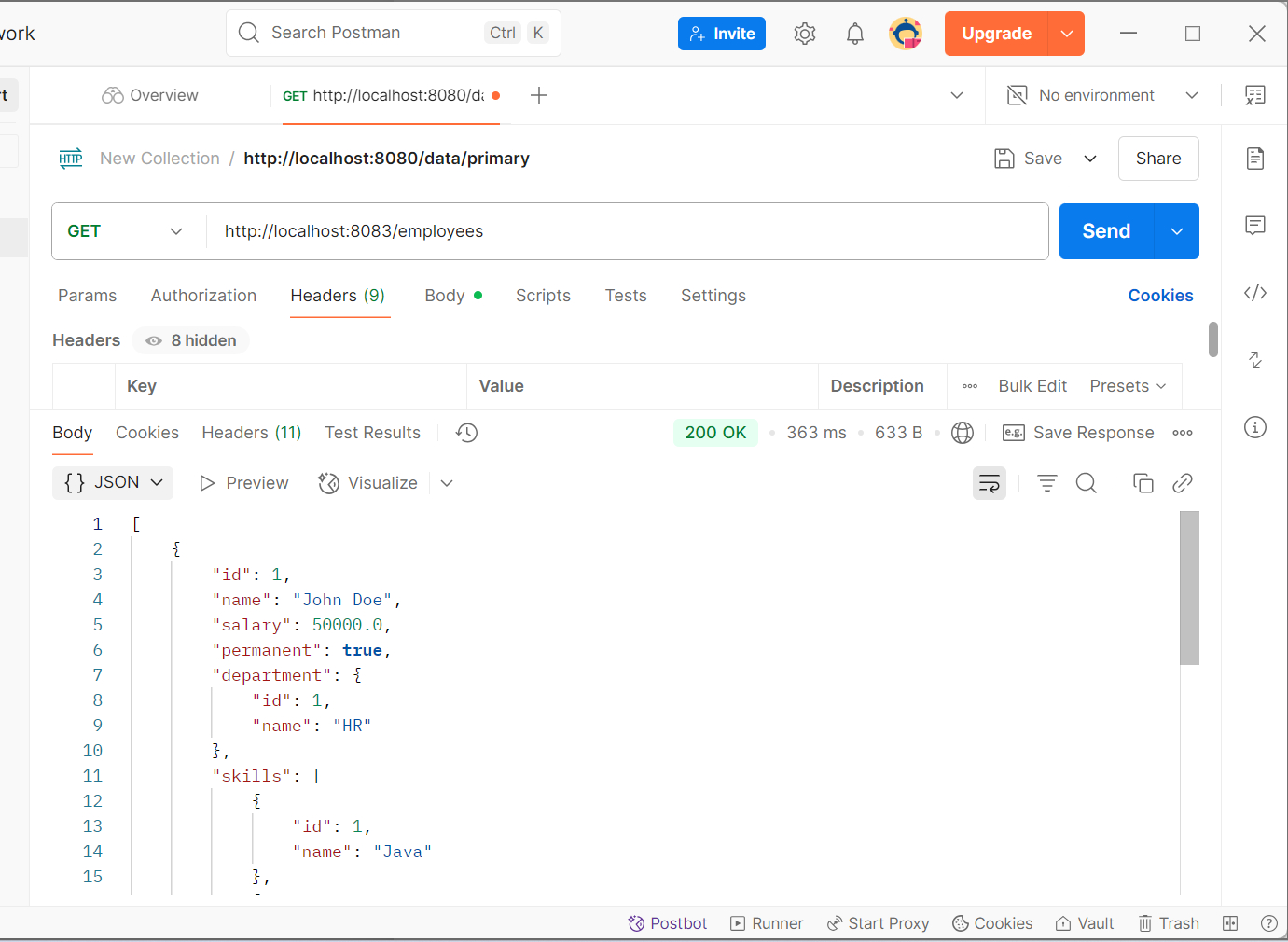
public String hello() {

return "Hello, this is a secured endpoint!";

}

}

**Output:**



**Creating Microservices for account and loan**

**Account**

**AccountController.java**

package com.cognizant.account;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/accounts")

public class AccountController {

@GetMapping("/{number}")

public Account getAccountDetails(@PathVariable String number) {

return new Account("00987987973432", "savings", 234343);

}

}

class Account {

private String number;

private String type;

private int balance;

public Account(String number, String type, int balance) {

this.number = number;

this.type = type;

this.balance = balance;

}

public String getNumber() { return number; }

public String getType() { return type; }

public int getBalance() { return balance; }

}

**Application.java**

package com.cognizant;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

**application.properties**

# account service uses default port 8080

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.5</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>account</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**Loan**

**LoanController.java**

package com.cognizant.loan;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/loans")

public class LoanController {

@GetMapping("/{number}")

public Loan getLoanDetails(@PathVariable String number) {

return new Loan("H00987987972342", "car", 400000, 3258, 18);

}

}

class Loan {

private String number;

private String type;

private int loan;

private int emi;

private int tenure;

public Loan(String number, String type, int loan, int emi, int tenure) {

this.number = number;

this.type = type;

this.loan = loan;

this.emi = emi;

this.tenure = tenure;

}

public String getNumber() { return number; }

public String getType() { return type; }

public int getLoan() { return loan; }

public int getEmi() { return emi; }

public int getTenure() { return tenure; }

}

**Application.java**

package com.cognizant;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

**application.properties**

server.port=8081

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.5</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>loan</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

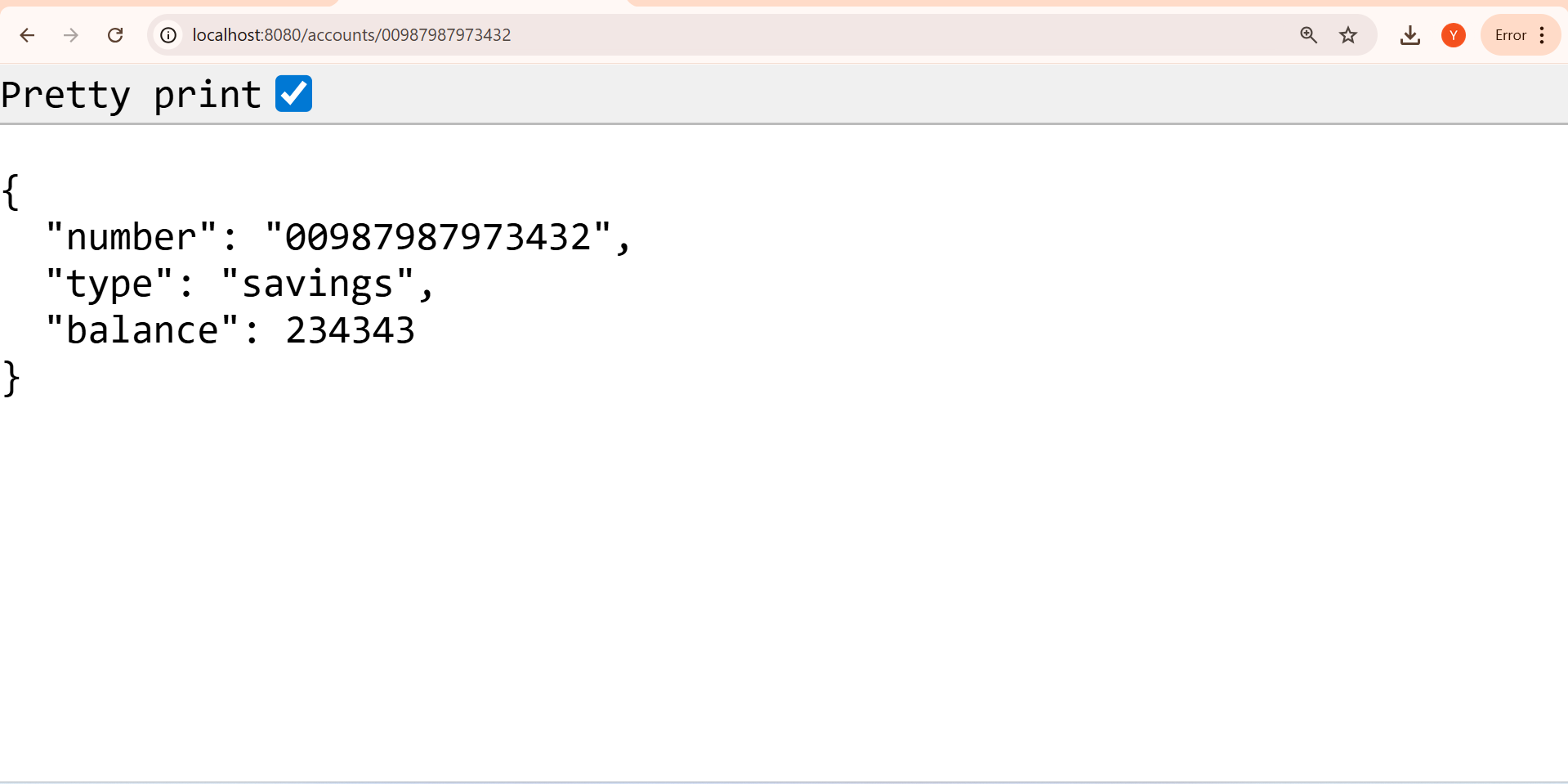
</plugins>

</build>

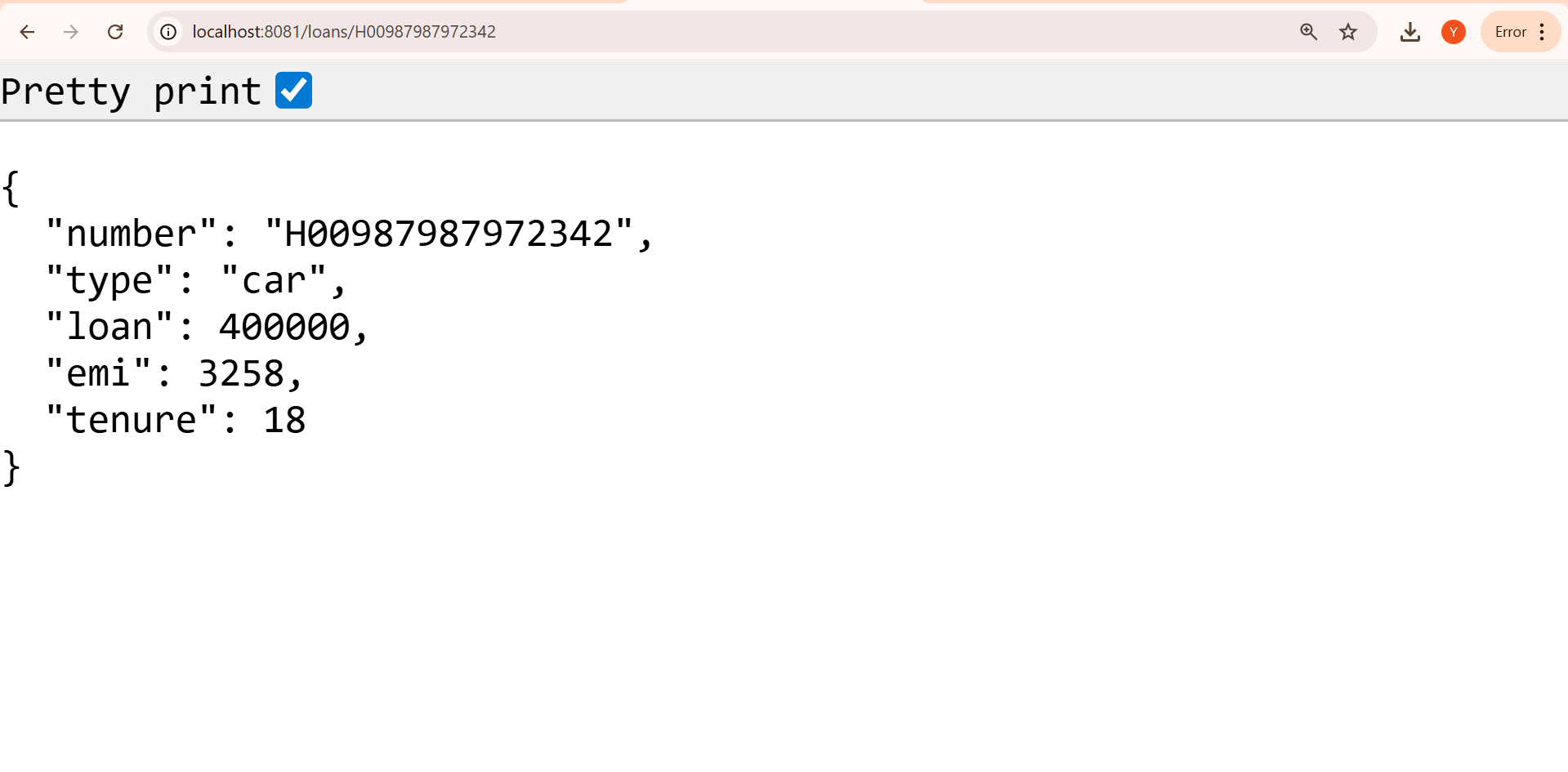
</project>

**Output:**

**Account:**



**Loan**



**Create Eureka Discovery Server and register microservices**

**eureka-discovery-server**

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>eureka-discovery-server</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>eureka-discovery-server</name>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>2022.0.3</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

</project>

**EurekaDiscoveryServerApplication.java**

package com.cognizant.eureka;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication

@EnableEurekaServer

public class EurekaDiscoveryServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaDiscoveryServerApplication.class, args);

}

}

**application.yml**

server:

port: 8761

eureka:

client:

register-with-eureka: false

fetch-registry: false

greetservice

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>greet-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>greet-service</name>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>2022.0.3</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

</project>

**GreetServiceApplication.java**

package com.cognizant.greet;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class GreetServiceApplication {

public static void main(String[] args) {

SpringApplication.run(GreetServiceApplication.class, args);

}

}

**GreetController.java**

package com.cognizant.greet;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class GreetController {

@GetMapping("/greet")

public String greet() {

return "Hello World";

}

}

**application.yml**

server:

port: 8082

spring:

application:

name: greet-service

eureka:

client:

service-url:

defaultZone: <http://localhost:8761/eureka>

api-gateway

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>api-gateway</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>api-gateway</name>

<description>Spring Cloud API Gateway</description>

<packaging>jar</packaging>

<properties>

<java.version>21</java.version>

<spring-cloud.version>2023.0.1</spring-cloud.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-gateway</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**ApiGatewayApplication.java**

package com.cognizant.apigateway;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

**application.yml**

server:

port: 9090

spring:

application:

name: api-gateway

cloud:

gateway:

routes:

- id: greet-service

uri: lb://GREET-SERVICE

predicates:

- Path=/greet-service/\*\*

filters:

- StripPrefix=1

eureka:

client:

service-url:

defaultZone: <http://localhost:8761/eureka>

**LogFilter.java**

package com.cognizant.apigateway.filter;

import org.springframework.cloud.gateway.filter.GatewayFilterChain;

import org.springframework.cloud.gateway.filter.GlobalFilter;

import org.springframework.core.Ordered;

import org.springframework.stereotype.Component;

import org.springframework.web.server.ServerWebExchange;

import reactor.core.publisher.Mono;

@Component

public class LogFilter implements GlobalFilter, Ordered {

@Override

public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {

System.out.println("Request received: " + exchange.getRequest().getURI());

return chain.filter(exchange);

}

@Override

public int getOrder() {

return -1;

}

}

**output:**

